We claim:

1. A system for performing variable time parallel processing on a plurality of ordered incoming entities, the system comprising:

a plurality of processors for processing the plurality of ordered incoming entities;

a processor scheduler coupled to the plurality of processors for assigning each of the plurality of incoming entities to one of the plurality of processors; and

a pre-processor coupled to the processor scheduler for keeping track of an order of the plurality of incoming ordered entities.

parallel processing 718/102

2. The system of claim 1 further comprising:

a plurality of memory blocks coupled to the plurality of processors, which the plurality of incoming ordered entities need to access.

3. The system of claim 1 further comprising:

an input buffer coupled to the plurality of processors, for buffering the plurality of ordered incoming entities.

4. The system of claim 1, wherein each of the plurality of ordered incoming entities comprises a data packet.

5. The system of claim 1, wherein each of the plurality of processors can run a plurality of contexts.

6. A method for performing variable time processes in parallel on a plurality of ordered incoming entities, on a plurality of processors, the method comprising:

709/234

assigning each of the plurality of incoming entities to one of the plurality of processors;

pre-processing each of the plurality of ordered incoming entities to establish an order of the plurality of ordered incoming entities; and

processing each of the plurality of ordered incoming entities on the corresponding one of the plurality of processors to which it is assigned.

7. The method of claim 6, wherein the plurality of ordered incoming entities comprise a plurality of packets in a network.

8. The method of claim 7 wherein the processing comprises:

extracting information from a header of each of the plurality of incoming packets;

hashing the extracted information for each of the incoming packets;

storing the hashed information; and

responsive to a new packet being received, comparing the hash of the extracted information for the new packet with the stored information.

9. The method of claim 6, wherein the assigning comprises:

selecting one of the plurality of processors which is free to process one of the plurality of ordered incoming entities at the time that the one of the plurality of ordered incoming entities is received.

10. The method of claim 6, wherein the pre-processing comprises:

determining whether a first in the plurality of ordered incoming entities is currently being processed at the time when a subsequent one of the plurality of ordered incoming entities is received;

responsive to determining that the first in the plurality of ordered incoming entities is currently being processed:

not starting processing of the subsequent one of the plurality of ordered incoming entities;

redetermining at a later time whether the first in the plurality of ordered incoming entities is currently being processed; and

709/234
Data flow
synchronization

responsive to determining that the first in the plurality of ordered incoming entities is not currently being processed, starting processing of the subsequent one of the plurality of ordered incoming entities.

11. A method for determining the optimal number of processors to be used in a processing unit for parallel processing of variable time processes on a plurality of packets in a network, in which each of the plurality of packets is received at the processing unit in one packet time, the method comprising:

solving $(A/N)^N = P$, for N, where A represents an average number of packet times that a processor takes to finish processing a packet, N represents the number of processors to be employed, and P represents a probability that N processors would not be sufficient for purposes of processing the received packets.

12. A method for determining the optimal number of processors to be used in a processing unit for parallel processing of variable time processes on a plurality of packets in a network, in which each of the plurality of packets is received at the processing unit in one packet time and each of the processors runs a plurality of contexts, the method comprising:

solving $[A/(N*C)]^{(N*C)} = P$, for N, where A represents an average number of packet times that a processor takes to finish processing a packet, C represents a number of the plurality of contexts on each of the processors, N represents the number of processors to be employed, and P represents a probability that N processors would not be sufficient for purposes of processing the received packets.

13. A method for determining the optimal number of processors to be used in a processing unit for parallel processing of variable time processes on a plurality of packets in a network, in which each of the plurality of packets is received at the processing unit in one packet time, and each of the processors runs a plurality of contexts, and has an input buffer with capacity to store "J" of the plurality of received packets, the method comprising:

solving $[A/(N*C)]^{(N*C)+J} = P$, for N, where A represents an average number of packet times that a processor takes to finish processing a packet, C represents a number of the plurality of contexts on each of the processors, N represents the number of processors to be employed, and P represents a probability that N processors would not be sufficient for purposes of processing the received packets.

14. A system for performing variable time processes in parallel on a plurality of packets in a network which comprise a plurality of flows, the system comprising:

a plurality of processors for processing the plurality of packets;

a processor scheduler coupled to the plurality of processors for assigning each of the plurality of packets to one of the plurality of processors; and

a pre-processor coupled to the processor scheduler for keeping track of the ones of the plurality of packets comprising each of the plurality of flows.

15. A system for performing variable time processes on a plurality of packets in a network in parallel, the plurality of packets comprising a plurality of flows, the system comprising:

a plurality of processors for processing the plurality of packets;

a processor scheduler coupled to the plurality of processors, for assigning each of the plurality of packets to one of the plurality of processors;

a pre-processor coupled to the processor scheduler for keeping track of the ones of the plurality of packets comprising each of the plurality of flows;

a plurality of memory blocks coupled to the plurality of processors, which the plurality of packets need to access; and

an input buffer coupled to the plurality of processors, for buffering the plurality of packets.

16. A method for performing variable time processes in parallel on a plurality of processors on a plurality of packets in a network which comprising a plurality of flows, the method comprising:

assigning each of the plurality of packets to one of the plurality of processors;

pre-processing each of the plurality of packets to keep track of the ones of the plurality of packets comprising each of the plurality of flows; and

processing each of the plurality of ordered incoming entities on the corresponding one of the plurality of processors to which it is assigned.

17. A method for performing variable time processes in parallel on a plurality of processors on a plurality of packets in a network which comprise a plurality of flows, the method comprising:

assigning each of the plurality of packets to one of the plurality of processors;

determining whether a first in the plurality of packets comprising one of the plurality of flows is currently being processed at the time when a subsequent one of in the plurality of packets comprising the one of the plurality of flows is received;

responsive to determining that the first in the plurality of packets is currently being processed:

not starting processing of the subsequent one of the plurality of packets;

redetermining at a later time whether the first in the plurality of packets is currently being processed;

responsive to determining that the first in the plurality of packets is not currently being processed, starting processing of the subsequent one of the plurality of packets; and

processing each of the plurality of packets on the corresponding one of the plurality of processors to which it is assigned.

18. An electronically readable medium storing a program for permitting a computer to perform a method for performing variable time processes in parallel on a plurality of ordered incoming entities, on a plurality of processors, the method comprising:

assigning each of the plurality of incoming entities to one of the plurality of processors;

pre-processing each of the plurality of ordered incoming entities to establish an order of the plurality of ordered incoming entities; and

processing each of the plurality of ordered incoming entities on the corresponding one of the plurality of processors to which it is assigned.

19. An electronically readable medium storing a program for permitting a computer to perform a method for determining the optimal number of processors to be used in a processing unit for parallel processing of variable time processes on a plurality of packets in a network, in which each of the plurality of packets is received at the processing unit in one packet time, the method comprising:

solving $(A/N)^N = P$, for N, where A represents an average number of packet times that a processor takes to finish processing a packet, N represents the number of processors to be employed, and P represents a probability that N processors would not be sufficient for purposes of processing the received packets.

20. An electronically readable medium storing a program for permitting a computer to perform a method for performing variable time processes in parallel on a plurality of processors on a plurality of packets in a network which comprise a plurality of micro-flows, the method comprising:

assigning each of the plurality of packets to one of the plurality of processors;

determining whether a first in the plurality of packets comprising one of the plurality of micro-flows is currently being processed at the time when a subsequent one of in

the plurality of packets comprising the one of the plurality of micro-flows is received;

responsive to determining that the first in the plurality of packets is currently being processed:

not starting processing of the subsequent one of the plurality of packets;

redetermining at a later time whether the first in the plurality of packets is currently being processed;

responsive to determining that the first in the plurality of packets is not currently being processed, starting processing of the subsequent one of the plurality of packets; and

processing each of the plurality of packets on the corresponding one of the plurality of processors to which it is assigned.